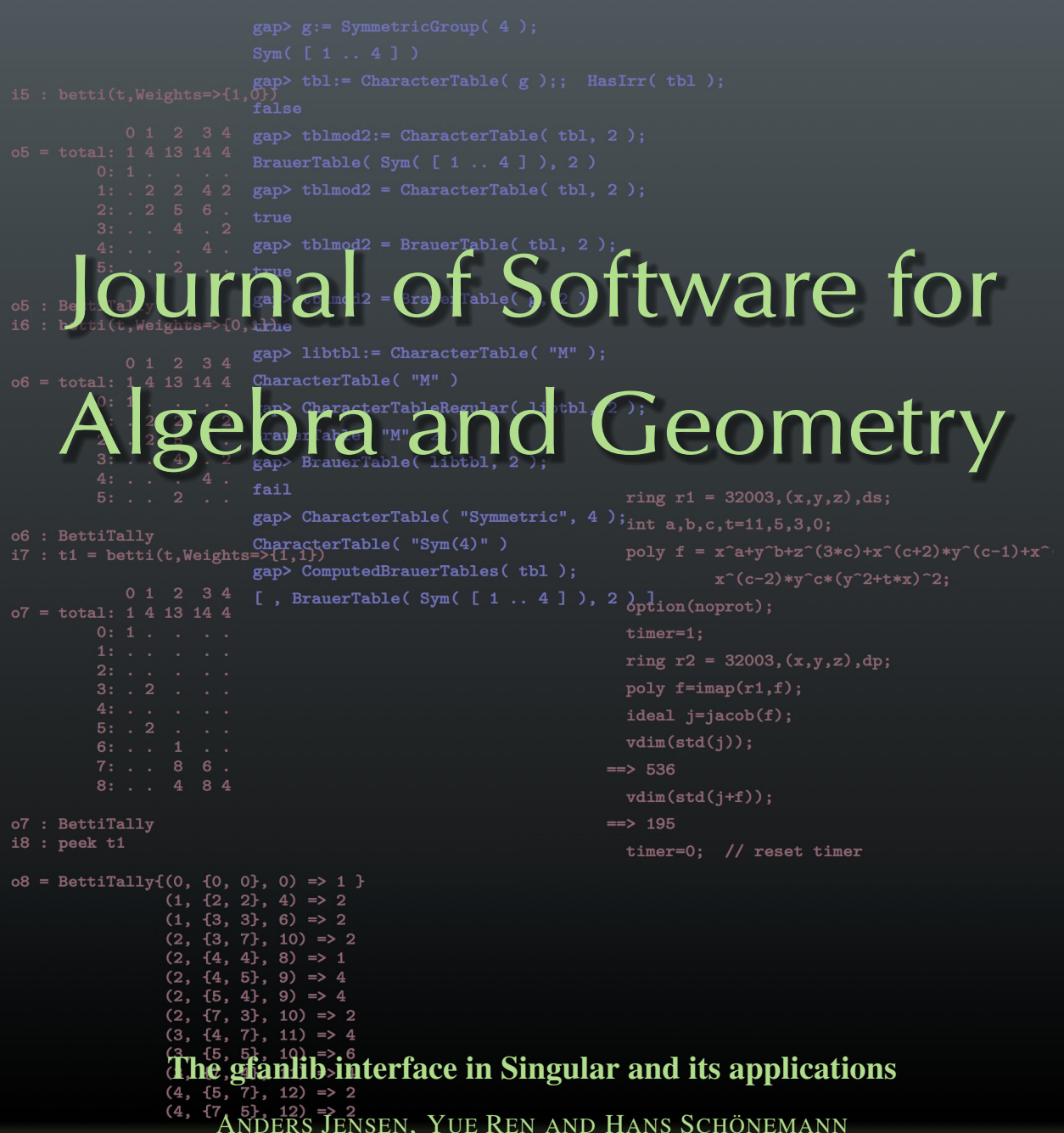

Journal of Software for Algebra and Geometry

The gfanlib interface in Singular and its applications

Anders Jensen, Yue Ren and Hans Schönemann

vol 9 2019

JSAG **9** (2019), 81–87
dx.doi.org/10.2140/jsag.2019.9.81

The Journal of Software for
Algebra and Geometry

The gfanlib interface in Singular and its applications

ANDERS JENSEN, YUE REN AND HANS SCHÖNEMANN

ABSTRACT: We briefly report on SINGULAR's low-level interface to GFANLIB: its usage on interpreter level, its implementation using SINGULAR's `blackbox` framework and two of its applications.

1. INTRODUCTION. SINGULAR [Decker et al. 2016] is a comprehensive computer algebra system for polynomial computations with particular emphasis on applications in algebraic geometry, commutative algebra and singularity theory. GFANLIB is a library derived from GFAN [Jensen 2011], a software package for computing tropical varieties and Gröbner fans by the first author. The GFANLIB interface allows SINGULAR users to work with polyhedral cones, polyhedral fans and operations thereon.

The goal of this paper is threefold: In Section 2, we provide an overview of the functionality in GFANLIB and how it can be accessed in SINGULAR. In Section 3, we describe the `blackbox`-framework in SINGULAR and illustrate how it can be used to introduce new types and functions to SINGULAR using the GFANLIB interface as an example. In Section 4, we showcase two applications of GFANLIB, demonstrating its use in high performance computations.

2. GFANLIB AND ITS INTERFACE IN SINGULAR. The class `ZCone` of GFANLIB encapsulates a rational polyhedral cone built on CDDLIB [Fukuda 2016] with arbitrary precision integral normal vectors. A `ZCone` object can be in one of four states (with an increasing level of knowledge), and it automatically advances its state if necessary:

(0) Defining equations and inequalities are known.

(1) A basis for the orthogonal complement is known. Its cardinality equals the codimension of the cone.

(2) A minimal set of inequalities has been determined. Each inequality cuts out a facet of the cone.

(3) The equations and inequalities from (1) and (2) have been written in a unique canonical form. This allows fast cone comparisons.

The class is accessible from SINGULAR as the type `cone`; see Figure 1. Cone A is defined as the non-negative span of the row vectors of the matrix M, $A = (\mathbb{R}_{\geq 0})^4 \cdot M$, while cone B is defined by considering the rows as inequalities, $B = (\mathbb{R}_{\geq 0})^2 \cdot M^t$. In both cases the cones are stored with an outer (H-) description, which means the first case requires a double description conversion. As a side effect, the inequalities that are obtained through the conversion are indeed facet normals and they are therefore marked as such.

MSC2010: primary 68W30; secondary 14L24, 14T05.
Keywords: Singular, gfanlib, interface.

```
> LIB "gfan.lib";
> intmat M[4][2]=(1,0),
  (1,1),(1,2),(1,3);
> cone A=coneViaPoints(M);
> A;
AMBIENT_DIM
2
FACETS
0, 1,
3,-1
> cone B=coneViaInequalities(M);
> B;
AMBIENT_DIM
2
INEQUALITIES
1,0,
1,1,
1,2,
1,3
```

Figure 1. A sample SINGULAR session using the type `cone`.

A `ZFan` class for representing polyhedral fans is provided. Cones can be inserted to such fan and when that happens, all faces are implicitly regarded as being inserted too. The `ZFan` class allows for easy conversion to ray incidence lists as in Figure 5.

Finally, the interface offers the possibility of computing mixed volume of Newton polytopes using [Jensen 2016], see Figure 2.

```
> LIB "gfan.lib";
> ring r=0,(x,y),dp;
> mixedVolume(list(x2y+1,xy2+1));
3
```

Figure 2. An example of computing mixed volume in SINGULAR.

3. THE BLACKBOX FRAMEWORK OF SINGULAR. Adding new data types and associated functions to SINGULAR used to require changes in many places due to the static nature of its type system. This process is now streamlined with the introduction of the type `blackbox`. An object of type `blackbox` is a `struct` of function pointers as explained in the following subsection. This is largely equivalent to defining new types as subclasses of an abstract class having the methods listed below being virtual, thereby taking advantage of inheritance in C++. With our solution on the other hand, we have full control and can add types at runtime as we like.

***Adding new data types to* SINGULAR.** Data types governed by `blackbox` are represented by a pair: a `void*` pointer to a call table, and an `int` serving as a unique type identifier. Registering a new type begins with filling a `struct` with function pointers to the following functions, of which the latter five are optional and come with preset defaults (see also Figure 3, lines 23–27):

(1) `blackbox_Init`: creates a default object
(2) `blackbox_destroy`: destroys an object (see Figure 3, lines 3–6)
(3) `blackbox_Copy`: copies an object (see Figure 3, lines 8–15)
(4) `blackbox_String`: converts an object to string
(5) `blackbox_Print`: prints an object (default: print conversion to string)
(6) `blackbox_Assign`: assign other types to object (default: raises error)
(7) `blackbox_OpX`: operations with object as first operand (default: raises error)
(8) `blackbox_serialize`: serialization for parallel computing (default: raises error)

(9) `blackbox_deserialize`: deserialization (default: raises error)

Next, that `struct` plus the name of the new type has to be passed to `setBlackboxStuff`, which then returns the unique type identifier (see Figure 3, line 31).

```
1  int coneID; // type identifier
2
3  void* bbcone_Init(blackbox* b)
4  {
5    return new gfan::ZCone();
6  }
7
8  char* bbcone_String(blackbox* b,
9                      void* d)
10 {
11   gfan::ZCone* zc =
12     (gfan::ZCone*) d;
13   string s = zc->toString();
14   return omStrDup(s.c_str());
15 }
```

```
18 void mod_init(SModulFunctions* p)
19 {
20   blackbox *b =
21   (blackbox*) omAlloc0(sizeof(*b));
22
23   b->blackbox_Init=bbcone_Init;
24   b->blackbox_Copy=bbcone_Copy;
25   b->blackbox_Assign=bbcone_Assign;
26   b->blackbox_destroy=bbcone_Destroy;
27   b->blackbox_String=bbcone_String;
28
29   [...]
30
31   coneID=setBlackboxStuff(b,"cone");
32 }
```

Figure 3. `blackbox` wrapper for type `gfan::ZCone`.

***Adding new functions to* SINGULAR.** Functions governed by `blackbox` require two `leftv` as input and a `BOOLEAN` as ouput (see Figure 4). The first `leftv` represents the return value of the function when called from the interpreter, the second `leftv` represents the input. The `BOOLEAN` that is returned signals the interpreter whether an error occured during the function call. Objects of type `leftv` are generic objects which can represent any object of any type that is available on the interpreter-level of SINGULAR. The unique type identifier of the input can be seen by calling `leftv->Typ()`, while the identifier of the output is to be stored in `leftv->rtyp`. The data of the input can be obtained by calling `leftv->Data()`, and the data of the output is to be stored in `leftv->data`. Moreover, `leftv->next` can be used to access the latter arguments of the input. It is set to `NULL` if the function has been called with one argument. To make the function callable from the interpreter, `iiAddCproc` needs to be called with:

(1) a `string` with the library containing documentation (empty string possible)

(2) a `string` with the desired function name on the interpreter level

(3) a `BOOLEAN` which marks the routine as static (`TRUE`) or global (`FALSE`)

(4) the function name on the kernel level

4. APPLICATIONS OF THE GFANLIB INTERFACE.

In this section, we briefly present two applications of SINGULAR's low-level interface to GFANLIB based on the previously described `blackbox` framework.

```
BOOLEAN facets(leftv res, leftv args)
{
  if ((args != NULL) && (args->Typ() == coneID) && (args->next == NULL))
  {
    gfan::ZCone* zc = (gfan::ZCone*) args->Data();
    gfan::ZMatrix zm = zc->getFacets();
    res->rtyp = BIGINTMAT_CMD;
    res->data = (void*) zMatrixToBigintmat(zm); // conversion function
    return FALSE;
  }
  WerrorS("facets: unexpected parameters");
  return TRUE;
}
void bbcone_setup(SModulFunctions* p)
{
  [...]
  p->iiAddCproc("gfan.lib","facets",FALSE,facets);
  [...]
}
```

Figure 4. `blackbox` wrapper for function `gfan::ZCone::getFacets()`.

Tropical varieties. Tropical geometry studies algebraic varieties based on what is commonly described as their combinatorial shadow. These so-called tropical varieties are the supports of finite polyhedral complexes.

Definition 1. Let K be a field with non-trivial valuation $\nu : K^* \to \mathbb{R}$ and residue field $\mathfrak{K}$. Fix a splitting $\nu(K^*) \to K^*$, writing t^a when referring to the image of $a \in \Gamma_\nu$. Given a polynomial $f = \sum_{\alpha \in \mathbb{N}^n} c_\alpha \cdot x^\alpha \in K[x]$ and an ideal $I \trianglelefteq K[x]$, we define for any weight vector $w \in \mathbb{R}^n$:

$$\mathrm{in}_w(f) = \sum_{w\cdot\alpha - \nu(c_\alpha)\ \text{max.}} \overline{t^{-\nu(c_\alpha)} c_\alpha} \cdot x^\alpha \in \mathfrak{K}[x] \text{ and } \mathrm{in}_w(I) = \langle \mathrm{in}_w(f) \mid f \in I \rangle \trianglelefteq \mathfrak{K}[x].$$

The tropical variety of I is then given by:

$$\mathcal{T}(I) = \overline{\{w \in \mathbb{R}^n \mid \mathrm{in}_w(I) \text{ monomial free}\}}.$$

Computing tropical varieties is an algorithmically challenging task, requiring sophisticated techniques from both computer algebra and convex geometry. The first techniques were developed for the field of complex Puiseux series $\mathbb{C}\{\{t\}\}$ [Bogart et al. 2007], in which the authors exploited that its uniformizing parameter t can be regarded as a variable of the polynomial ring, allowing them to rely on classical Gröbner bases techniques. For more general fields with valuation, a new theory of Gröbner bases taking the valuation on the field into account was introduced [Chan and Maclagan 2019].

In contrast, the implementation in SINGULAR is based on recent results [Markwig and Ren 2019], which describe tropical varieties over valued fields K with tropical varieties over any dense subring of the ring of integers $\mathcal{O}_K \subseteq K$. The practical advantage of this approach is that it is compatible with existing standard bases techniques in SINGULAR.

Example 2. Consider the ideal $I := \langle x_1 - 2x_2 + 3x_3, 3x_2 - 4x_3 + 5x_4 \rangle \trianglelefteq \mathbb{Q}[x_1, \ldots, x_4]$ and the 2-adic valuation on $\mathbb{Q}$. Figure 5 shows input and output for computing the tropical variety in SINGULAR.

```
> LIB "tropical.lib";
> ring r = 0,x(1..4),dp; number p = 2;
> ideal I = x(1)+2*x(2)-3*x(3), 3*x(2)-4*x(3)+5*x(4);
> tropicalVariety(I,p);

RAYS                                MAXIMAL_CONES
-2 -1 1 -1 1\# 0                     {0 1}    \# Dimension 3
-1 1 -1 1 -1\# 1                     {0 2}
0 -3 1 1 1\# 2                       {0 4}
0 1 -3 1 1\# 3                       {1 3}
0 1 1 -3 1\# 4                       {1 5}
0 1 1 1 -3\# 5
```

Figure 5. SINGULAR output for Example 2.

The output is a polyhedral fan in $\mathbb{R}^5$ whose intersection with the affine hyperplane $\{e_1 = 1\}$ yields a polyhedral complex covering the tropical variety. Hence the rays #2–#5 represent vertices at infinity so that the last four maximal 2-dimensional cones are in fact unbounded edges; see Figure 6.

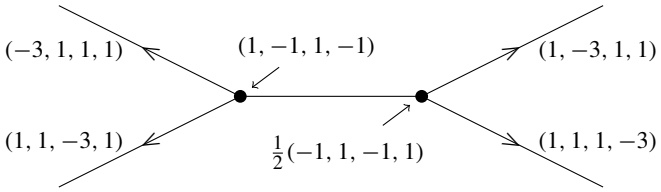

Figure 6. $\mathcal{T}(I)$ for Example 2.

GIT-fans. GIT-fans describe the variation of possible GIT-quotients [Dolgachev and Hu 1998]. A case of particular importance is the action of an algebraic torus on an affine variety, for which explicit algorithms exist [Berchtold and Hausen 2006; Keicher 2012; Boehm et al. 2016]. All algorithms have been implemented in the SINGULAR library `gitfan.lib` and are publicly available as part of the official SINGULAR distribution.

Example 3. [Boehm et al. 2016, Example 5.2]
The Cox ring of Grass(2, 5) is isomorphic to $\mathbb{C}[T_1, \ldots, T_{10}]/\mathfrak{a}$, where the ideal $\mathfrak{a}$ is generated by the following Plücker relations and the $(\mathbb{C}^*)^5$ action on Grass(2, 5) is induced by the following grading matrix Q with respect to which $\mathfrak{a}$ is homogeneous:

$$\begin{aligned} &T_5T_{10} - T_6T_9 + T_7T_8,\\ &T_1T_9 - T_2T_7 + T_4T_5,\\ &T_1T_8 - T_2T_6 + T_3T_5,\\ &T_1T_{10} - T_3T_7 + T_4T_6,\\ &T_2T_{10} - T_3T_9 + T_4T_8 \end{aligned} \qquad Q := \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & -1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The natural S_5-symmetry on Grass(2, 5) acts, up to sign, as the following permutations on the variables:

$$S_5 = \langle (2,3)(5,6)(9,10), (1,5,9,10,3)(2,7,8,4,6) \rangle \leq S_{10}.$$

The GIT-fan consists of 76 cones, which decompose into 6 orbits under the S_5 action. Figure 7 shows the computation of the GIT-fan without and with symmetry. The GIT-fan is generated by 76 maximal cones, which decompose into 6 orbits under the group action.

```
> LIB "gitfan.lib";
> ring R = 0,T(1..10),dp;
> ideal J =
.   T(5)*T(10)-T(6)*T(9)+T(7)*T(8),
.   T(1)*T(9)-T(2)*T(7)+T(4)*T(5),
.   T(1)*T(8)-T(2)*T(6)+T(3)*T(5),
.   T(1)*T(10)-T(3)*T(7)+T(4)*T(6),
.   T(2)*T(10)-T(3)*T(9)+T(4)*T(8);
> intmat Q[5][10] =
.   1, 1, 1, 1, 0, 0, 0, 0, 0, 0,
.   1, 0, 0, 0, 1, 1, 1, 0, 0, 0,
.   0, 1, 1, 0, 0, 0, -1, 1, 0, 0,
.   0, 1, 0, 1, 0, -1, 0, 0, 1, 0,
.   0, 0, 1, 1, -1, 0, 0, 0, 0, 1;
> list S5 = G25Action();
```

```
> S5;
[1]:
   |  1  2  3  4  5  6  7  8  9 10|
[...]
[120]:
   |  1  2  3  4  5  6  7  8  9 10|
   | 10  9  7  4  8  6  3  5  2  1|
> fan SigmaIgnoringSymmetry =
.   GITfan(J,Q);
> fVector(SigmaIgnoringSymmetry);
1,20,110,240,225,76
> fan SigmaModuloSymmetry =
.   GITfan(J,Q,S5);
> fVector(SigmaModuloSymmetry);
1,16,56,67,33,6
```

Figure 7. SINGULAR input and output for Example 3.

Figure 8 shows the adjacency graph of the maximal cones and their S_5 orbits.

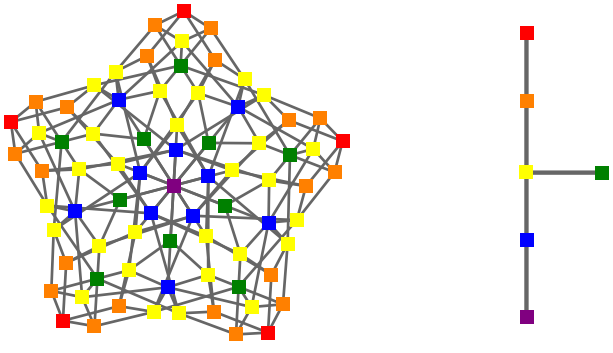

Figure 8. Adjacency graph of the maximal cones in the GIT-fan of Grass(2, 5) and of their S_5-orbits.

SUPPLEMENT. The online supplement contains source code for the GFANLIB interface to SINGULAR.

REFERENCES.

[Berchtold and Hausen 2006] F. Berchtold and J. Hausen, "GIT equivalence beyond the ample cone", *Michigan Math. J.* **54**:3 (2006), 483–515. MR

[Boehm et al. 2016] J. Boehm, S. Keicher, and Y. Ren, "Computing GIT-fans with symmetry and the Mori chamber decomposition of $\bar{M}_{0,6}$", 2016. arXiv

[Bogart et al. 2007] T. Bogart, A. N. Jensen, D. Speyer, B. Sturmfels, and R. R. Thomas, "Computing tropical varieties", *J. Symbolic Comput.* **42**:1-2 (2007), 54–73. MR

[Chan and Maclagan 2019] A. J. Chan and D. Maclagan, "Gröbner bases over fields with valuations", *Math. Comp.* **88**:315 (2019), 467–483. MR

[Decker et al. 2016] W. Decker, G.-M. Greuel, G. Pfister, and H. Schönemann, "SINGULAR 4-1-0: a computer algebra system for polynomial computations", 2016, available at http://www.singular.uni-kl.de.

[Dolgachev and Hu 1998] I. V. Dolgachev and Y. Hu, "Variation of geometric invariant theory quotients", *Inst. Hautes Études Sci. Publ. Math.* 87 (1998), 5–56. With an appendix by Nicolas Ressayre. MR

[Fukuda 2016] K. Fukuda, "cddlib reference manual, cddlib Version 094h", 2016, available at https://www.inf.ethz.ch/personal/fukudak/cdd_home.

[Jensen 2011] A. N. Jensen, "Gfan 0.5, a software system for Gröbner fans and tropical varieties", 2011, available at http://home.math.au.dk/jensen/software/gfan/gfan.html.

[Jensen 2016] A. N. Jensen, "An implementation of exact mixed volume computation", pp. 198–205 in *Mathematical software—ICMS 2016*, Lecture Notes in Comput. Sci. **9725**, Springer, [Cham], 2016. MR

[Keicher 2012] S. Keicher, "Computing the GIT-fan", *Internat. J. Algebra Comput.* **22**:7 (2012), 1250064, 11. MR

[Markwig and Ren 2019] T. Markwig and Y. Ren, "Computing tropical varieties over fields with valuation", *Found. Comput. Math.* (online publication August 2019).

RECEIVED: 11 Jul 2018 ACCEPTED: 19 Nov 2018

ANDERS JENSEN:

jensen@imf.au.dk
Aarhus Universitet, Aarhus, Denmark

YUE REN:

yue.ren@mis.mpg.de
Max Planck Institute for Mathematics in the Sciences, Leipzig, Germany

HANS SCHÖNEMANN:

hannes@mathematik.uni-kl.de
Technische Universität Kaiserslautern, Kaiserslautern, Germany

JOURNAL OF SOFTWARE FOR ALGEBRA AND GEOMETRY vol 9, no 1, 2019

1948-7916(2019)9:1;1-D